А. В. Акименко

ВИЗУАЛИЗАЦИЯ ТРЁХМЕРНЫХ СЦЕН МЕТОДОМ ТРАССИРОВКИ ПУТИ С ИСПОЛЬЗОВАНИЕМ ГРАФИЧЕСКОГО API VULKAN

Рассматриваются принципы и особенности реализации алгоритма трассировки пути (path tracing) для фотореалистичной визуализации трёхмерных сцен с использованием низкоуровневого графического API Vulkan. Анализируются преимущества Vulkan для задач рендеринга с интенсивными вычислениями, в частности, возможности расширений для трассировки лучей (Vulkan Ray Tracing).

Введение

Создание фотореалистичных изображений является одной из фундаментальных задач компьютерной графики. Трассировка пути, основанная на алгоритме Монте-Карло для решения уравнения рендеринга, позволяет достичь высокой степени реализма за счёт точного моделирования глобального освещения. Однако этот метод традиционно связан с высокими вычислительными затратами, что ограничивало его применение в приложениях реального времени.

С появлением современных графических процессоров (GPU) и низкоуровневых API, таких как Vulkan, открылись новые возможности для ускорения сложных алгоритмов рендеринга. Vulkan предоставляет разработчикам прямой контроль над оборудованием, минимизируя накладные расходы драйвера и позволяя эффективно использовать параллельные вычислительные мощности GPU. Особый интерес представляют стандартизированные расширения Vulkan Ray Tracing, которые обеспечивают аппаратную поддержку для ускорения операций пересечения лучей с геометрией [1].

Целью данной работы является анализ подходов к реализации рендеринга методом трассировки пути с использованием возможностей графического API Vulkan, в частности его расширений для трассировки лучей, и обсуждение ключевых аспектов построения соответствующего конвейера визуализации.

I. Основы реализации трассировки лучей в Vulkan: Структуры и Конвейер

Реализация трассировки пути средствами Vulkan, особенно с использованием специализированных расширений, включает несколько ключевых этапов. Создание Структур Ускорения (Acceleration Structures - AS). Эффективная трассировка лучей невозможна без пространственных структур данных, ускоряющих поиск пересечений лучей с объектами сцены. Расширение VK KHR acceleration structure определяет два типа AS:

- Структура ускорения нижнего уровня (Bottom-Level Acceleration Structure - BLAS): Строится для каждой уникальной геометрии (меша) в сцене;

– Структура ускорения верхнего уровня (Top-Level Acceleration Structure - TLAS): Строится над экземплярами (instances) BLAS, представляя всю сцену и учитывая их трансформации.

Создание и обновление AS – вычислительно интенсивные задачи, которые выполняются на GPU с использованием команд Vulkan. Качество и скорость построения AS напрямую влияют на общую производительность рендеринга.

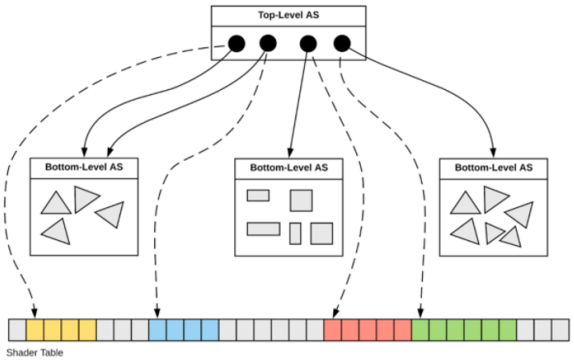

Рис. 1 – Обзор двухуровневой структуры ускорения

Построение Конвейера Трассировки Лучей. Расширение VKKHRraytracingpipeline вводит новый тип графического конвейера, отличный от традиционного растеризационного. Этот конвейер включает специализированные программируемые шейдерные стадии:

– Ray Generation (RGen): Шейдер генерации лучей. Запускается для каждого пикселя изображения, генерирует первичные лучи (например, из камеры). Он инициирует процесс трассировки вызовом vkCmdTraceRaysKHR;

– Intersection (Isect): Шейдер пересечения (опциональный). Используется для проверки пересечения луча с пользовательскими примитивами (не только треугольниками);

– Any Hit (AHit): Шейдер любого пересечения (опциональный). Вызывается при каждом обнаружении пересечения луча с примитивом. Может использоваться, например, для обработки полупрозрачных объектов (alpha testing), решая, нужно ли продолжать поиск более близких пересечений;

– Closest Hit (CHit): Шейдер ближайшего пересечения. Вызывается для самого близкого к источнику луча пересечения. Здесь реализуется основная логика path tracing: определение свойств материала поверхности, расчет отражений/преломлений

(на основе BRDF/BSDF), генерация вторичных лучей, сэмплирование источников света и накопление цвета вдоль пути луча;

– Miss (Miss): Шейдер промаха. Вызывается, если луч не пересек ни один объект в сцене. Обычно используется для выборки цвета из карты окружения (environment map) или задания фонового цвета.

Таблица Привязки Шейдеров (Shader Binding Table - SBT). SBT – это область памяти на GPU, которая связывает геометрические примитивы в структурах ускорения с соответствующими группами шейдеров (Hit Group – комбинация Isect, AHit, CHit) и шейдерами Miss. При вызове vkCmdTraceRaysKHR GPU использует индексы из SBT для определения, какие шейдеры выполнять при пересечении луча с конкретным объектом или при промахе. Это обеспечивает гибкость в назначении различных шейдерных программ разным объектам сцены.

II. Реализация алгоритма и практические вызовы

Реализация Алгоритма Трассировки Пути. Основная логика самого алгоритма path tracing реализуется преимущественно в шейдерах Ray Generation и Closest Hit. RGen инициирует трассировку для каждого пикселя. CHit рекурсивно (или итеративно) продолжает путь луча после каждого отскока от поверхности, пока не будет достигнута максимальная глубина трассировки, луч не уйдет в фон (обработается Miss шейдером) или его вклад не станет пренебрежимо малым (например, с использованием техники "русской рулетки"). На каждом шаге происходит накопление световой энергии вдоль пути.

Несмотря на аппаратное ускорение, реализация эффективной трассировки пути в реальном времени остается сложной задачей. Основные вызовы включают:

– Шум: Стохастическая природа метода Монте-Карло приводит к появлению шума на изображении, особенно при малом количестве сэмплов (лучей) на пиксель. Требуются методы шумоподавления (denoising), которые могут быть интегрированы в конвейер рендеринга [1];

– Производительность: Сложность сцены, количество источников света, свойства материалов и глубина трассировки сильно влияют на производительность. Необходимы оптимизации, такие как важностное сэмплирование (importance sampling) источников света и BRDF, адаптивная глубина трассировки, оптимизация структур ускорения;

– Управление ресурсами: Vulkan требует явного управления памятью, ресурсами (буферы, изображения, AS, SBT) и синхронизацией операций на GPU, что усложняет разработку по сравнению с API более высокого уровня.

III. Выводы

Графический API Vulkan, особенно с его расширениями для трассировки лучей, предоставляет мощную и гибкую основу для реализации алгоритмов глобального освещения, таких как трассировка пути. Низкоуровневый доступ к GPU и аппаратное ускорение ключевых операций позволяют достигать высокой производительности, открывая перспективы для использования фотореалистичного рендеринга в интерактивных приложениях и играх.

Хотя остаются вызовы, связанные с управлением шумом, оптимизацией производительности и сложностью разработки на Vulkan, дальнейшее развитие аппаратного обеспечения, алгоритмов шумоподавления и совершенствование инструментов разработки делают трассировку пути с использованием Vulkan все более привлекательным подходом для достижения нового уровня визуального качества в компьютерной графике реального времени. Дальнейшие исследования могут быть направлены на интеграцию более сложных моделей материалов, гибридные подходы (сочетание растеризации и трассировки) и адаптивные методы сэмплинга.

1. Andersson, E. Real-Time Path Tracing using Vulkan Ray Tracing. Master's Thesis, Linköping University, Department of Computer and Information Science. - 2021 [Электронный ресурс]. – Режим доступа: http://liu.diva-portal.org/smash/get/diva2:1572197/FULLTEXT01.pdf – Дата доступа: 30.03.2025.
2. Vulkan Ray Tracing Specification. Khronos Group. [Электронный ресурс]. – Режим доступа: https://registry.khronos.org/vulkan/specs/1.3-extensions/man/html/VKKHRraytracingpipeline.html – Дата доступа: 30.03.2025.
3. Kajiya, J. T. The rendering equation // ACM SIGGRAPH Computer Graphics. – 1986. – Vol. 20. – №. 4. – Pp. 143-150.

Акименко Александр Владимирович, студент кафедры программного обеспечения и информационных технологий БГУИР, alexander0aki@gmail.com.

Научный руководитель: Красковский Павел Николаевич, старший преподаватель кафедры программного обеспечения и информационных технологий БГУИР, kraskovskij@bsuir.by.